LPIC 3 –Security

Study Guide/Notes v1

This guide is based on Sander Van Vugt LPIC 303 Security Course and resources from Internet and My Mind. All the credit goes to Sander, Google and me in some Parts.

Hope you like it !

The book is not intended to be a definitive guide, but a way to guide you through the topics to get you familiarized with the subject. Use it with another resource like Man pages. Also i will complement the book in the future.

Who am i?

My name is Joao Paulo de Andrade Filho, a 21 years old guy who likes to improve the others learning. I think that knowledge should be free and the LPIC 3 is a complex test, so if you struggle in some parts dont be sad, you will make it !

I made this book to get you technically capable to handle issues with Linux Security in General.

This is my first book, so if you want to give ideas or ways for me to improve, you can send me an email:

andradejp@nullsec.com.br

How this book is organized?

We have 4 chapters:

1- Operating System General Security
2- Managing Linux User Security
3 - Securing Linux Services
4 - Securing Linux Infrastructure

They cover almost every subject in the LPIC 3 303 test, some additional things you will find in man pages. That is, you need to read the pages if you have any doubt. Or you can contact me, feel free : -) .

Operating System General Security

First thing to make system secure -> **Patches**
How to make an update strategy?
-> Visit resources on the net, ex CVE or Linux Distro Site etc
-> Management system, **ubuntu landscape, redhat satellite**

Commands Related to updates:

yum updateinfo -> consult the repository and provide a report about security notices
yum updateinfo list -> provide a list of everything that is avaiable plus status
yum updateinfo advisoryid -> Search information regarding an ID

*** The id is found in the updateinfo command, as a number like this: RHSA-2018:0188**

On ubuntu:

apt-get -s dist-upgrade -> shows a list of all updates available for the system
apt-get -s dist-upgrade | grep ¨^Inst¨ | grep -i secur -> searches for security updates
apt-get -s dist-upgrade | greo ¨^Inst¨ | awk -F ¨ ¨ {print $2 } | xargs apt-get install -> install security patches manually.
unattended-upgrades -> install all the updates automatically on Ubuntu
****Landscape** -> paid service to manage upgrades.

Validate packages

rpm -Va -> verify all

On Ubuntu:

debsums -l -> generate a list of MD5
debsums -c -> verify all checksums

Encrypted Block Device

LUKS -> Encryption Layer

Steps to make an encrypted device:

1- **/dev/sdb1** (Raw Device)
2- **cryptsetup luksFormat /dev/sdb1**(creates the encryptionLayer)
3 - **cryptsetup luksOpen /dev/sdb1 secret (Secret is the name of the device)**
4 - **mkfs.ext4 on /dev/mapper/secret**
***The file system must be created on the mapper**
 5 - **mount /dev/mapper/secret /mnt**
6- **umount /mnt**
7- **crypsetup luksClose /dev/mapper/secret**

Mount persistently

It needs a luks Key (to not enter the passphrase while mounting)
***Store the key on another device to be more secure**

Important Files:

Automate the luks open -> **/etc/crypttab**
Automate the mount -> **/etc/fstab**

How to create a key file:

dd if=/dev/urandom of=/root=lukskey bs=4096 count=1
chmod 600 /dev/lukskey
cryptsetup luksAddKey /dev/sdb1 /root/luksKey -> Add the key to the encrypted device

The crypttab file:

vim /etc/crypttab

Syntax:

Name of the encrypted device / Name of the original device / Name of the Key

Example:
- > **secret /dev/sdb1 /root/luksKey** (or none to prompt for a password)

The fstab file :

vim /etc/fstab
-> **/dev/mapper/secret /secret ext4 defaults 0 0**

Security Related Mount Options

Mount with specific options (Split in different partitions)

/tmp
/var
/var/log
/usr
/home

Options:

-> **nodev** -> no devices can be accessed from this filesystem
-> **nosuid** - > No SUID programs may be running
-> **noexec** -> Prevent users from running executables
To apply those changes, write it in fstab.

Monitor filesystem changes

AIDE (Advanced Instrusion Detection Enviroment)
/etc/aide.conf -> Config file with variables to set database locations
aide --init -> Create a database of the system (Move the database to another location for better security)
/var/lib/aide -> location of the database
aide --check -> check the database to see if changes are detected.
***AIDE its meant to be used in static parts of the filesystem because it checks for filesystem changes.**

Securing the grub bootloader:

When a system boots:

1- Post
2- Bootable device
3- Bootloader (grub2 or something like)
4 - Kernel + Initramfs
5 - Services
6 - Shell

***In a security perspective, it can go wrong in 2, because an intruder can boot from an USB device.**

There are 2 types of passwords for bootloader:

Global -> It Makes impossible to boot
OS password - > Secure one OS

How to apply passwords:

/etc/grub.d/01_users
    set superusers=¨linda¨
    password linda somepw
    password lisa anotherpw (not a superuser, it will not have power to make   modifications)

To make her capable to modify, define a menuentry (**take care to not make mistakes, if its the case, grep menuentry on /boot/grub2/grub.cfg then paste and modify**)
/etc/grub.d/40_custom
    menuentry ... --users lisa
    {

```
        set root(hd0,msdos2)
        linux /vmlinuz ...
        initrd ...
    }
```

After all, write modifications to /boot/grub2/grub.cfg:
- > **grub2-mkconfig -o /boot/grub2/grub.cfg**

***In grub1, specify password somepw within a menuentry or outside**
***There is no way to define users accounts in grub 1**

Modifying text console security

There are 2 issues we can handle:

1) Display messages at console
2) Deal with control-alt-del

The first:

/etc/issue or /etc/issue.net -> display before login, example: SSH BANNER
/etc/motd -> display messages after login (Message of the day)
~/.hushlogin -> the contents of motd will not be shown for an specific user

Second issue:

In some servers, if we press ctrl alt delete the server will restart.

To solve this problem:

systemctl status ctrl-alt-del.target
Systemctl disable ctrl-alt-del.target
systemctl mask ctrl-alt-del.target

Modifying graphical console security

/etc/gconf -> defaults for graphical desktop
/gconf/schemas -> schemas for options in desktop gconftool-2 -> tool to modify gconf options

Configuring system logging

Who is logging?

Services (We can configure to feed syslog) **Syslog**
Systemd-journald (Feeds syslog)

/var/log -> Contains the logs of the system

***Syslog logs using facilities like news, cron etc**

Originally:

Syslog:
    facility.priorities - destination
Rsyslog -> **Work with modules**

Important informations:
/etc/rsyslog.conf -> Configuration file
IM -> input modules
OM -> output modules(Send the output of log to somewhere else)

***The file can have rules too.**

At the end of file, we can enable remote loggin:

***.* @@remote-host:514** --> Every facility and every priority will be sent to remote host (@@ stands for tcp)

journalctl -> Tool to see journal logs

Configure remote logging

Secure remote logging will use TLS

Requisites:

->Time sync

->Certificates (certtool)

->Port 6514 TCP

On the client:

gtls driver must be confingured

To configure:

yum install rsyslog-doc

yum install elinks

cd /usr/share/doc/rsyslog*/ -> documentation

systemctl status chronyd -> Chronyd is the time server for RHEL 7

Preparing the key:

certtool -> utility to create CA

certtool --generate-privkey --outfile ca-key.pem -> generate the key

chmod 400 ca-key.pem

certool --generate-self-signed --load-privkey ca-key.pem --outfile ca.pem

-> generate the ca certificate

certtool --generate-privkey --outfile server1-key.pem --bits 2048
certtool --generate-request --load-privkey server1-key.pem --outfile server1-request.pem -> Generate the request
certtool --generate-certificate --load-request server1-request.pem --outfile server1-cert.pem --load-ca-certificate ca.pem --load-ca-privkey ca-key.pem
-> Generate the certificate

On server1:

mkdir /etc/rsyslog-keys cd /etc/rsyslog.d/

vim logserver.conf

```
    $DefaultNetstreamDriver gtls
    $DefaultNetstreamDriverCAFile /etc/rsyslog-keys/ca.pem
    $DefaultNetstreamDriverCertFile /etc/rsyslog-keys/server1-cert.pem
    $DefaultNetStreamDriverKeyFile /etc/rsyslog-keys/server1-key.pem

    $ModLoad imptcp
    $InputTCPServerStreamDriverMode 1
    $InputTCPServerStreamDriverAuthMode anon (Doesnt require the client to authenticate )
    $InputTCPServerRun 6514
```

systemctl restart rsyslog
systemctl status rsyslog
yum install gnutls -> to have support for tls

*** These options are fairly intuitive so i did not explain them.**

On workstation

scp server*.pem ipserver1:/etc/rsys-keys vim /etc/hosts
ip workstation.example.com workstation ip server1.example.com server1
scp /etc/hosts ipserver1:/etc/hosts mkdir /etc/rsyslog-keys

cp ca.pem /etc/rsyslog-keys/
vim /etc/rsyslog.d/log-client.conf
```
    $DefaultNetStreamDriverCAFile /etc/rsyslog-keys/ca.pem
    $DefaultNetStreamDriver gtls
    $ActionSendStreamDriverMode 1
    $ActionSendStreamDriverAuthMode anon
    *.* @@(o)server1.example.com:6514
```
-> @@ for tcp. (o) for tls

yum install rsyslog-gnutls
systemctl restart rsyslog
logger HELLO FROM WORKSTATION -> To see if works

Configure advanced filtering

****Used to filter only important messages**

There are 2 types of filtering:

1) Traditional
facility.priority - destination

2) Property based filters
Work with:
Properties

Fixed:
from host: messages:

Comparison operators:
contains
startswith
isequal

After that, send to a Destination

Configuring:

cd /etc/rsyslog.d/
vim remotefilter.conf :
:fromhost, isequal, ¨workstation.example.com¨ /var/log/server1/messages
:fromhost, isequal, ¨workstation.example.com¨ ~ (~ is the exclude mark to prevent the log to be full of local and remote messages).
systemctl restart rsyslog logger HELLO AGAIN

Managing log rotation

cd /etc/cron.daily/logrotate -> script to rotate logs on a daily basis
/etc/logrotate.conf -> files with rotation options
/etc/logrotate.d/ -> specific logrotate files

Making journald logs persistent

Originally :
Journald logs to /run/log/journal
Default -> **syslog   /var/log**
In many distros journald logs to syslog too

To make persistent:
mkdir -p /var/log/journal
systemctl restart systemd-journald
***Its important to watch the size of this file and create a log rotation.**
***Journald cant do remote logging**
/etc/systemd/journald.conf -> used to configure parameters for journald.

Using logwatch for log analysis

It runs from crond

/etc/cron.daily/0logwatch -> file to config logwatch on a daily basis

***By default the root will receive a mail from logwatch**

/etc/logwatch/conf/logwatch.conf -> configuration file for logwatch

***To pick an example of an logwatch config, go to /usr/share/logwatch and copy one**

logwatch --range all(the default range is yesterday) -> get every log

Reading the audit log

Proccess: **auditd**

/var/log/audit/audit.log -> Audit Log (hard to read)

/etc/audit/auditd.conf -> main configuration file of audit

Audit Configuration Client/Server

On the client:

Auditd process running -> the process can send messages to syslog on the server or in auditd in the server.

If we want to send messages directly to syslog, we need to create:

/etc/audisp/plugins.d/syslog.conf and set it to **active=yes**

If we want to send messages to an auditd process running on the server, we need to create:

/etc/audisp/plugins.d/au-remote.conf and set it to **active=yes**

/etc/audisp/audisp-remote.conf and set a **remote_server = server.example.com (Example)**

On the server:

If we want to receive messages for auditd, we need to set a **tcp_listen_port = some number** in **/etc/audit/auditd.conf**

yum search audisp -> plugins to audit (remote)

yum install audisp-plugins

Audit reporting

ausearch -> searches for specific events in the audit file

aureport -> generic reporting utility

autrace -> Can be used to create an audit trace

-> Searching using ausearch

ausearch -i -a 87(event code)

-> Searching for an event (using autrace)

autrace/sbin/useradd laura -> make a trace of this command

Writing Custom Audit Rules

Utility to use: **auditctl**

***Ordering does matter, if a rule is matched, the above rules will not be matched**

auditctl -w(watch) /etc/passwd -p wa(write and atribute changes) -k user-edit(custom key)

→ in this rule, after create an user we can grep user-edit in audit log

auditctl -w /etc/sysconfig/ -p rwa(read, write and attr changes) -k sysconfig-changes

→ Same thing as above

auditctl -w /bin -p x(execution of binaries) -k binari-exec

→ Watch for binary execution

auditctl -a exit,always(trigger a syscall when exit) -F arch=b32 -F arch=b64 -S rename -S renameat -k rename

auditctl -a exit,alway -F dir=/home/ -F uid=0 -C auid!=obj_uid
→ log every time when root access files in home

auditctl -l -> overview of rules
auditctl -D -> remove the rules

****We can copy the contents of auditctl -l and paste in /etc/audit/audit.rules to make them persistent**

Using predefined audit sets

/usr/share/audit -> files with rules use, we can copy them to /etc/audit/audit.rules and use
Rotation of audit files -> auditd.cron file

Audit keystroke logging

Edit the **/etc/pam.d/system-auth** and include the line:
session required pam_tty_audit.so enable=root -> Enable keystroke logging for root.
To see the report, type: **aureport --tty**

Understanding MAC

A type of access control that is controled by the system, not by the user. The kernel implement the policy(rules) and there is nothing a user can do against it.

Some types of MAC and their differences:

- > SE Linux
Consists of many rules
Rules stored in policy and applied to the filesystem
Starts with an enviroment where everything is closed
Used by red hat and open SUSE
Created by NSA

-> AppArmor (SUSE and Ubuntu) Easy
Works with profiles
Starts with an enviroment where everything is open

- > SMACK
Easy
Embedded linux devices

Configuring App Armor

Too see if its running: **systemctl status apparmor**
/etc/apparmor.d/ -> directory with profiles

To create profiles:

1-Find the path to command
2- **aa-genprof /bin/dd** (You need to pass the complete path to the command like this example.)
3- Create events(Actions using the command) with the command
4- Scan system for apparmor events (****The Glob means allow everything in the directory**)
5- We can also edit profile by hand

/sys/kernel/security/apparmor/ -> runtime information of what apparmor is doing

To see if a program is enforced by apparmor, check its pid, go to **/proc/pid/attr ; cat current**

ps -Zaux -> Z is an option related to security
aa-compain /bin/dd -> Back learning mode
aa-enforce -> enforce apparmor
aa-disable -> disable apparmor

Understanding SE Linux

When the system starts the kernel is loaded and the kernel loads selinux
/etc/sysconfig/selinux -> Config file to put modes for selinux (disabled, enforcing, permissive(only logs, not enforce anything))

****We can toogle enforcing/permissive with setenforce command**

Every source and target object in the system has a context, these contexts are defined in rules in selinux policy and those rules define what is allowed or not allowed

Configuring selinux file context

sestatus -> shows the status of selinux
ls -lZ -> Shows the selinux context labels
ps Zaux -> Show selinux context labels for processes
netstat -Ztulpen -> same thing for ports
id -Z -> same for user accounts

semanage -> allows to do alot of options as change file contexts, port and so on
semanage fcontext -a -t httpd_sys_content_t ¨ /net(/.*)? ¨ -> apply this context to all files in / net

**** If you want to change the context of another files based on the first file, first we need to see the context of the original file then we apply the right context to the file of our choice.**

restorecon -Rv /net -> Restore the conf to the directory (Apply them)
chcon -> another bin to apply context (write to the filesystem, not to the system, never use it)
semanage fcontext -l -> list all contexts
sepolicy manpage -a -> write policy to the man pages

Configuring selinux port context

**** For troubleshooting, first set selinux to permissive (setenforce permissive) to see if the selinux if blocking, then, if it is blocking, write the right selinux port context to make it available**

semanage port -a -t ssh_port_t -p tcp 4824 -> change the port context, allowing ssh to bind to port 4824 on this example.
semanage port -m -t ssh_port_t -p tcp 4824 -> modify a port that is already defined
semanage boolean -l -> list on and of switches for selinux (Booleans)
setsebool -P ftpd_anon_write on -> -P make it persistent, -p not. This

command set this boolean to on .

Analyzing seliux events

/var/log/audit/audit.log | grep -i AVC(Access vector cache) -> grep selinux messages
/var/log/messages | grep sealert -> same thing as above, but more easy (use sealert command to describe the alert)
audit2allow -> command that pick messages from audit log and allow them
semodule -i mypol.pp -> load the module

Configuring custom rules and modules

semodule -l -> shows the modules

Policy file consists of 2 files:
.te file -> contain the rules
.pp file -> module file

Module file (Example):

```
module ssh_http ;
require {
    type sshd_t;
    type http_port_t;
    class tcp_socket{name_bind} ;
} ;
allow sshd_t http_port_t:tcp_socker{name_bind};
```

/etc/selinux/targeted/modules/active/modules -> location of modules (targeted)

chckmodule -M -m -o ssh.mod ssh.te -> create the module file

semodule_package -o ssh.pp -m ssh.mod -> create the pp file semodule -i **ssh.pp** -> insert the module

Managing users in selinux

semanage user -l -> list the users of selinux
seinfo -a selinux_unconfined_type -x -> shows what is avaiable in unconfined type semanage login -a -s user_u lisa -> add user lisa to user_u roll

SMACK (Simplified mandatory access control)

Used in embbeded devices
Talk to the kernel and use an init script

Understanding Linux Kernel Architecture

The kernel is on the Ring0 and the user space on Ring3. A process can reach in ring0 by using **syscalls.**
Programs have their memory spaces individually and a program cant access another program´s shared memory. There is an excesssion if the program use IPC or MMAP.

To get maximum process isolation, we can use:

-> Chroot
-> Containers
-> Cgroups

Fixing kernel security issues

Risks:

1- Buffer Overflow
Protection: App must have limited restricted memory, for the kernel: patches and security updates

2- Privilege Escalation
Protection: Disable SUID where it is not needed, su and sudo shells too

3- Rootkit
Protection:
Run filesystem integrity checker
No kernel modules, some rootkits can relay in certain kernel modules, if we disable then, they will not relay in anything

Fixing linux kernel vulnerabilities

/proc/sys/kernel ; echo 1 > modules_disabled -> disable modules(to load again, the system need to be rebooted)
/proc/sys/kernel ; echo 2 > randomize_va_space -> Enable randomization
grep nx /proc/cpuinfo -> see if the nx feature is enabled in the computer

Managing User Linux Security

Summarizing basic Permission Usage

Permissions		
	Files	Dir
read	read	ls
write	modify	create/delete
execute	run	cd

Managing permissions:

chown -> change the owner
chgrp -> change the group
chmod -> change the permission mode

Managing Special Permissions

Permissions		
	Files	Dir
SUID	Run file as owner	-
GUID	Run file as group owner	Inherit group owner
Sticky Bit	-	Remove only if owner

Finding files with special permissions

find /usr/bin -perm 4000 -> find files only with this permission
find /usr/bin -perm -4000 -> match files with permission and anything else
find /usr/bin -perm /4000 -> match all files with permission and anything else and it permits to add permissions to search like 6000 (4 + 2)

Managing Default Permissions

To see default and apply default permissions:
→ **umask**

Default values:
Files: 666
Dir: 777

Umask calc:
umask 027
Files: (666 - 027) = 640
Dir: (777 - 027) = 750

To see the default umask, type **umask**
To change umask, type umask and choose the value, like umask 027
***To set an specific umask, put it in the .bashrc of an user.**
***The umask is set in bashrc and profile, to be persistent**

Managing access control lists

With acl we can assign multiple users and groups to a file and give permissions

getfacl -> get file acl that are currently applied
setfacl -> sets an acl
setfacl -m g:account:rw sales/ -> Sets the rw permission for the group

account the the sales directory (**Apply just in the dir and not in new itens**)

To set an ACL to new itens, we need to set a default ACL:
→ **setfacl -m d:group:account:rw sales/**

Adding support for ACLsl:
mount | grep ext -> see options to check if acl is present
tune2fs -l drive -> list options to see built in options (In ext4 acl is built in)
tune2fs -o acl drive -> add acl support
/etc/fstab -> add option acl to add acl support
***** Those options are for ext4 filesystem**

Using extended attributes

There are 2 types of attributes:
→ Regular
•lsattr -> list file attributes
•chattr -> change file attributes Ex: chattr +i file
chattr -i file

→ Extended
Allows users and admins to store aditional information in the file that can not be stored in the inode

Format: namespace.atribute
-security -> used by linux kernel security modules (SE Linux)
-system -> Used to store ACL
-trusted -> Can only be used by processes that are root
-user -> used to store any other type of data

getfattr -d /home -> dump the extended attributes of the directory
getfattr -m pattern /home -> search and attibute based on a pattern

-> Setting attributes
Utility to use: **setfattr**
setfattr -x -> remove attributes

Managing password properties

When creating an user, it is written in 2 files:

/etc/passwd
/etc/shadow -> place to store passwords (Encrypted)

There are also a file that we can change account and use of password properties:

/etc/login.defs

chage -> used to change age and password information
passwd -> used to change password related options
chage lisa -> Interactive mode to change passwd properties
echo passwd | passwd --stdin lisa -> change the passwd in one run
*** If we use the -1 option setting the expiration time, it will not expire**

Auditing User Accounts

getent passwd -> query passwd
getent passwd | cut -d : -f3 | sort -n | uniq -d -> sort double id user accounts

Understanding PAM

/etc/pam.d/ -> directory of pam
/lib/security -> directory of modules
ldd $(which login) -> see if the login command have libpam enabled.

Syntax of pam.d files

```
auth      required     pam_env.so
account     requisite
password   sufficient
session optional
```

/etc/securetty -> disable tty´s and other terminals
man -k pam -> see the pam modules. Important
/usr/share/doc -> documentation of pam (SAG in linux stands for system administrator guide)

*** We can add auth required pam_securetty.so within su module in pam.d to prevent certain terminal to use the su command**

pam_cracklib -> allows to check password security properties
pam_unix -> pam unix security properties
pam_deny -> denies everything
pam_allow -> allows everything

Configuring PAM

pam_time -> Account related module,
pam_limits -> set account limitations
/etc/security/limits.conf -> include ulimit alike functions in pam
/etc/security/time.conf -> allows to specify time ranges for certain users to log in the system

Applying account lockout with pam tally

Module that allows restrictions to the amount of login atempts
Implemented in /etc/pam.d/system-auth or login

Add above system auth(inside system-auth file) to deny after 3 tries
-> auth required pam_tally.sodeny=3 unlock_time=180 quiet

On red hat we can see the log in the **/var/log/secure.**

pam_tally2 -> also a command to see failures
pam_tally2 --reset -u lisa -> reset the counter of failures for user lisa

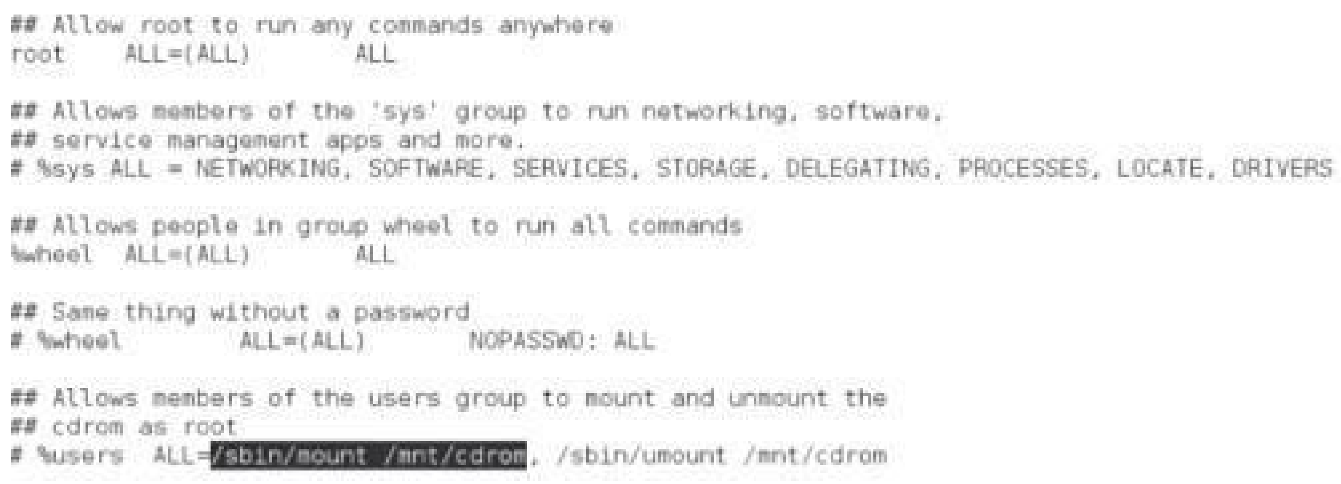
Configuring sudo

We can use **sudo -i or sudo su - to elevate privileges** to root, but it is not recommended, instead we will use every command with sudo.

visudo -> on redhat, edit the sudo file
etc/sudoers or /etc/sudoers.d -> sudo file on ubuntu

**** Define commands to users or %groups with names in the format:**
Cmd_Alias NETWORKING = /sbin/mount /mnt/cdrom
%admins ALL = NETWORKING
joao ALL = NETWORKING

Example:



```
## Allow root to run any commands anywhere
root    ALL=(ALL)       ALL

## Allows members of the 'sys' group to run networking, software,
## service management apps and more.
# %sys ALL = NETWORKING, SOFTWARE, SERVICES, STORAGE, DELEGATING, PROCESSES, LOCATE, DRIVERS

## Allows people in group wheel to run all commands
%wheel  ALL=(ALL)       ALL

## Same thing without a password
# %wheel        ALL=(ALL)       NOPASSWD: ALL

## Allows members of the users group to mount and unmount the
## cdrom as root
# %users  ALL=/sbin/mount /mnt/cdrom, /sbin/umount /mnt/cdrom
```

Understanding LDAP

The Lightweight Directory Access Protocol is an open, vendor-neutral, industry standard application protocol for accessing and maintaining distributed directory information services over an Internet Protocol (IP) network. (Font: Wikipedia)

CN = Common Name
DC - Domain component

Understanding Kerberos

Used to protect passwords while in transit and it can be used to authenticate users and applications against KDC

3 components
-> **KDC** - Key distributions center, hands out keys to the other participants, such as: Application server

Client user

The starting point:

Realm -> Auth domain, normally it reflects the dns name in uppercase
The principle in kerberos identify each participant, each principle have a password and a role assigned , two principles can be assinged to the same user example:
linda@example.com -> ordinary user
linda/admin@example.com -> user linda with admin credentials assigned to it Services can have principles as well, **if a service have a principle, it is stored in the krb5.keytab file.**

kinit -> utility that a user can use to initialize a new session klist -> check currently avaiable kerberos credentials.

kutil -> display the content of keytab files

Installing freeipa server Components:

→ LDAP Server

→ Kerberos (enhancing ldap)

→ DNS

→ Certificate system

→ NTP

** Install FreeIPA on a dedicated server to avoid conflicts

ipa-server-install --setup-dns --idstart=5000(user id of ipa server) --idmax=50000 -> Install the ipa and DNS

**** If the IPA complains about enter ad ip address, cancel the installation, go to /etc/hosts and create a new entry like with your IP, like 192.168.1.58 ipa.example.com**

ipactl -> manage services envolved in ipa

Installing FreeIpa Client

First add the free ipa dns in the client (sysconfig network scripts) **ipa-client-install** -> install the ipa in client

Understanding the role of SSSD

Central server in the auth proccess that is going to determine how the auth is going to happen

Method to access the server:

/etc/sssd/sssd.conf -> config file
/etc/krb5.conf -> kerberos config file

Managing user and groups in FreeIPA

We can add user and groups in the web interface

ipa help commands -> help for ipa commands ipa user-add karen -> add an user
ip user-mod karen --password -> set a new password ipa user-find karen -> find information about an user

Setting policies on FreeIPA

We can set policies in the web interface

Configuring centralized sudo rules

We can also set policies on the web interface

After the creation of the keytab in web interface:

ipa-getkeytab -s ipa.example.com -p nfs/server1.example.com -k /etc/krb5.keytab -> On the client, to request it

Securing Linux Services

Understanding DNS Security Issues

-Spoofing -> Someone pretends to be a valid name server
-Compromisse of the named daemon
-Unauthorized access solutions
-DNSSEC
-TSIG
-Restrict Access
-Run bind in chroot

-Securing DNS Generic Parameters

-> allow-query
-> allow-transfer
-> recursion
-> listen-on

Example:
acl ¨internal¨ { 192.168.4.0/24 ; } ;
allow-query{internal ; } ;

/etc/named.conf -> main config file.

TSIG -> Secure the communication between master and slave, works with a shared key that must be installed on each machine, also we need NTP.
DNSSEC -> Cryptographic security applied to dns

dnssec-keygen -a RSASHA1 -b 1024 -n ZONE myzone.example.com -> create a keypair
dnssec-keygen -a RSASHA1 -b 4096 -n KSK myzone.example.com -> create the secure entry key
cat Kmyzone*.key >> myzone.example.com -> write the key to myzone
dnssec-signzone -e +3024000 -N INCREMENT myzone.example.com -> sign the zone
After that, include the myzone.example.com.signed :

```
zone ¨myzone.example.com¨
{ type master ;
file ¨/var/named/master/myzone.example.com.signed¨;
allow-update { none ; } ;
} ;
```

After that, restart the named and contact the registrar to make sure that the keys are available to the registrars as well.

Securing apache

Security Challenges:
-> Scripts (Need to be confined in a very secure way)
-> Encryption (Add TLS)
-> Directory Specification
-> Auth

MAC is important to apache, make sure selinux or apparmor rules are used for apache

Security Options:
<Directory specification> </Directory>
Option Indexes -> make apache follow indexes
AllowOverride none -> allow or disallow htaccess

Order statements
<Directory /web> order allow, deny
allow from example.com   -> We can allow access from ips, websites and so on.
</Directory>

Authentication:
htpasswd -c joao /etc/passwd -> Create the user for authentication

<Directory /web/secret >
AuthName ¨Secret¨
AuthType basic
AuthUserFile /etc/.htpasswd
Require valid-user
</Directory>

Use SSL security (mod_ssl)

/etc/httpd/conf.d/ssl.conf -> ssl related parameters
SSLCertificateFile -> Certificate for SSL
SSLCertificateKeyFile -> Key for SSL
SSLCACertificateFile -> Refer to a non default CA file
/var/www/cgi-bin -> Directory where scripts are stored

Securing Email Threats:

-> DoS Attacks
-> Spam
-> Viruses

/etc/postfix/master.cf -> definition of subprocesses started by postfix
/etc/postfix/main.cf -> Postfix config file
postconf -> give access to all config parameters

Important parameters:

myhostname -> name of machine
myorigin -> defines where the message is comming from
mydestination -> tells which is the current destination
inet_interfaces -> interfaces for postfix
my_networks -> defines the networks where postfix is getting messages from
disable_vrfy_command -> make certain attacks impossible
smtp_helo_required -> make postfix more secure
smtpd_recipient_rest -> accept messages from anybody if is not set
in_flow_delay -> delay before messages are accepted
local_destination_concurrency ->max amount of messages that can be delivery to a local recipient locally
/etc/postfix/access -> used to put access restrictions (Ordering Does Matter)
Example:
john@somewhere.com REJECT
*@spammer.com REJECT
10.3 REJECT 10 ACCEPT

postmap /etc/postfix/access -> activate the access (convert access table to access.db)
**** By default, postfix uses unencrypted traffic on port 25 tls_key_file**

tls_cert_file -> Cert file
tls_tls_CA file -> CA File
tls_loglevel -> Define the log level
smtpd_use_tls -> enable tls

Securing SSH

Important options (Server Side) :

/etc/ssh/ssh_config -> client config file
/etc/ssh/sshd_config-> server config file Important options on server
Port 22 -> port specification
Protocol 2 -> Protocol section, use the 2 to be more secure
PermitRootLogin -> Disable or enable root login (Disable)
AllowUsers joao -> Allow users to log in ssh
PubkeyAuthentication -> permit the auth with keys
PasswordAuthentication -> Enable password authentication
GSSAPIAuthentication yes -> Enable kerberized authentication
StrictModes yes -> permissions that are set on keys, if set to yes auth will fail if there is a problem with too much permission on public private keys that have been created
SyslogFacility AUTHPRIV -> Sets the log level

Important options on client:

Protocol
Ciphers
Macs

**** We can specify specific user configs in .ssh folder in home directory**
ssh-keygen -t dsa -> generates a DSA Key
ssh-copy-id server1.example.com -> copy the public key to server1 (authorized keys)

ssh-agent /bin/bash -> start ssh agent for bash
ssh-add after the previous command -> add the key to ssh agent (On this session, we will not need to type passwords anymore)

ssh -L 888:example.com:80 -Nf server1.example.com -> configure local port forwarding, using ssh in the middle to connect to the site, now we can connect to local host on port 8888 and we will be redirected to example.com

Securing VSFTPD Problems

→ Weak auth
→ Easily hijacked
→ Plain text passwords (Use sftp to be more secure)

/etc/vsftpd/vsftpd.conf -> Config files

Important options (Default values, change it based on your needs, most of the are self explanatory) :

anonymous_enable=YES
local_enable=YES
write_enable=YES
local_umask=022
anon_upload_enabled=YES
anon_mkdir_write_enable=YES
xferlog_enable=YES
chown_uploads=YES -> Use file like a dropbox where a user that dropped the file cant access it anymore
idle_session_timeout
data_connection_timeout
chroot_local_user=YES

Understanding NFSV4 Security improvements

NFS ->Used to create a share on one server and access on the another server. With the same credentials on those servers.

Security problems:

-> Host Based
-> Uid Mapping
-> Random firewall ports
->Portmapper-> randomly assign ports when nfs is starting
->Services running as root
->Complex Portmapper, rpc.mountd, rpc.statd and lockd

Nfsv4
→ Random firewall ports * Fixed by using tcp port 2049
→ Host based security * Enhanced by kerberized authentication
→ Portmapper * Doesnt have to be used

Configuring Nfsv4 server and clients

File to configure shares -> **/etc/exports**
Syntax -> **share host(* for everything)**

Ex:
```
/etc/exports:
      /data     *(rw,no_root_squash)
```

no_root_squash(not recommended) -> **If the user root comes in the client, it will be the user root in the nfs share**

showmount -e localhost -> show nfs shares on localhost
mount server1:/data /mnt -> Mount the share on the client

***Without the no_root_squash, if the mount the share with the root user, we cant do anything, because the root user will be mapped to the nfs user nobody (without rights).**

Nfs authentication mechanism

/etc/exports (Default security=sys)
/data *(rw)

Using nfsv4 pseudo filesystem

mount server1:/   /mnt -> Instead of mounting every file, we can mount a common mount point (/) and it will mount all   the shares.

Nfsv4 ACL´s

->Not supported on redhat

nfsv4_acl -> list acls
nfsv4_editfacl -> edit an acl
nfsv4_getfacl -> get acl
nfsv4_setfacl -> command to set acl
Aren´t even implemented on every linux system

Example of an acl:
nfsv4_setfacl A::alice@nfsdomain.org:rxtncy -> alice has the equivalent read and execute permissions (rxtncy) and A for allow.

Understanding and using cifs unix extensions

Specific Samba options for lpic 303 (smb.conf):

unix extensions(G)

This boolean parameter controls whether Samba implments the CIFS UNIX extensions, as defined by HP. These extensions enable Samba to better serve UNIX CIFS clients by supporting features such as symbolic links, hard links, etc... These extensions require a similarly enabled client, and are of no current use to Windows clients.

Default: **unix extensions = no**

Understanding and configuring cifs security modes (ntlm, Kerberos)

Security option is in the general option within the smb.conf.
Default is **security = user.**
Also we have ads(Active Directory security) or domain security.

Managing, mapping and handling of cifs, acls and sids

getcifsacl -> get cifs acl
setcifsacl -> set cifs acl

Configure Free Radius to authenticate network nodes

Radius -> Remote authentication dial in user server

/etc/raddb/ -> Conf directory, radius db.
clientsf.conf -> security related settings, for every client we need to specify a config here.

Identifying and dealing with rogue router advertisements and Dhcp

/proc/sys/net/ipv6/conf/all -> Set forwarding to 0
accept_ra 0 (accept Router advertisements set to 0)

Configuring and operating openvpn server and clients

Port 1194.
adapters: tun and tap.

Configuring and operating IPsec Server and Clients

IPSEC - Vpn technology , security applied to the ip layer.
Software: Openswan or strongswan

on /etc/sysctl.conf:
net.ipv4.ip_forward = 1
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.send_redirects = 0

Also we need to open ports 500 udp and 4500 tcp/udp

```
iptables -t net -A POSTROUTING -s 192.168.4.0/24 -d 192.168.5.0/24 -j SNAT
--to 10.0.0.10
```

strongswan/ipsec.conf -> conf for strongswan ipsec

Securing Linux Infrastructure

Using tcpdump and wireshark

tcpdump -i eth0 -> on the interface

tcpdump -i eth0 -w file.pcap -> forward the traffic to a file

tcpdump -n -i eth0 -> it doesnt show names, only ips. (Doesnt resolve)

tcpdump -tttt -n -i eth0 -> make time more readable (not in epoch)

tcpdump -tttt -n -i eth0 port 22 -> filter to port 22

tcpdump -w ssh.pcap -i eth0 dst 192.168.1.10 port 22 -> target and ip to analyze ssh To read the file, we can use wireshark.

Introducing nmap

nmap -sS ip -> stealth scan

nmap -sV ip-> scan version

nmap -sn 192.168.1.0/24 -> scan network nmap -v -A 192.168.1.0/24 -> verbose all scan

nmap -PN ip -> scan to go through firewall (Try to pass) nmap -O ip -> OS scan

*** XMAS and FIN scans are also types of nmap scan.**

Nmap graphical interface -> zenmap

Understanding Nessus

Vulnerability Scanner Versions:

Cloud Version

Manager

Professional

Use Agents in the target that we are scanning (Better, but we didnt use it) OpenSource Alternative -> **openVAS**

Using tripwire

Versions:

Open Source -> Host intrusion detection system * **Alternative AIDE** Commercial -> Network intrusion detection system ***Alternative SNORT or Suricata**

Introducing SNORT

Network instrusion detection system * Paid
Free Version with almost no signature files. (Useless) **Alternative: SURICATA**

Performing a basic snort configuration

2 files to download:

snort*
daq*

snort -dvi eth0 -> Scanning for network traffic
/etc/snort/snort.conf -> config for snort
/etc/snort/rules/ -> rules for snort

Understanding Snort Output

Output goes by default in **/var/log/snort** Binary Files

snort -r to read files.

Burnyard 2 -> Tool to read snort binaries.

Introducing Nagios

-> Characteristics

Monitoring System
Web-based
Plugin capable.

htpasswd -c /etc/nagios/passwd nagiosadmin -> Create a passwd for nagios
After that, go to the ip of the server in browser

/etc/nagios/objects/ -> Objects basically are scripts to monitor specific itens

Using NTOP

Default port: **3000**
Web browser interface

Using John the ripper password cracking utility

john --test -> Test drive for john
unshadow /etc/papsswd /etc/shadow > /mypasswd.txt -> redirect the content of those 2 files to a single file
john mypasswd.txt -> john tries to crack the file created before

Introducing Puppet

Configuration management system

-> Characteristics

Puppet manages the current state of the SO
Works with manifest(puppet program that tells puppet what to do) with .pp extensions Works with modules (collection of manifests)
puppet apply -> apply the manifest file
puppet.conf -> Configuration file

-> Understanding iptables basics

Characteristics:

Ordering does matter
Policy -> Default action of a chain (Behavior)

Syntax:

Iptables -t TABLE -A CHAIN [-i/-o iface] [-s/-d addr] -p udp --sport/--dport 80 -j ACTION
Iptables -P(Policy) INPUT DROP -> Set the default action to a policy

Tables:

1. Filter Table
Filter is default table for iptables. So, if you don't define you own table, you'll be using filter table. Iptables's filter table has the following built-in chains.

INPUT chain – Incoming to firewall. For packets coming to the local server.
OUTPUT chain – Outgoing from firewall. For packets generated locally and

going out of the local server.
FORWARD chain – Packet for another NIC on the local server. For packets routed through the local server.

2. NAT table
Iptable's NAT table has the following built-in chains.

PREROUTING chain – Alters packets before routing. . This is used for DNAT (destination NAT)..e Packet translation happens when the packets are leaving the system. This helps to translate the source ip address of the packets to something that might match the routing on the desintation server.
POSTROUTING chain – Alters packets after routing. This is used for SNAT (source NAT).
OUTPUT chain – NAT for locally generated packets on the firewall.

3. Mangle table
Iptables's Mangle table is for specialized packet alteration. This alters QOS bits in the TCP header. Mangle table has the following built-in chains.

PREROUTING chain
OUTPUT chain
FORWARD chain
INPUT chain
POSTROUTING chain

4. Raw table
Iptable's Raw table is for configuration excemptions. Raw table has the following built-in chains.

PREROUTING chain
OUTPUT chain

iptables -L -> List rules iptables -F -> Flush Rules
iptables -D chain number -> delete rule
iptables -nL --line-numbers -> Line numbers

-Advanced iptables usage

iptables -A INPUT -m(loads an specific kernel module) state --state ESTABLISHED,RELATED - j ACCEPT
iptables-save > /root/iptablesRules -> save iptables rules
iptables -A INPUT -j LOG -> Log messages (/var/log/messages) **iptables-restore < /root/iptablesRules** -> restore rules

Working with firewalld

Work with objects like zones, objects and ports.

firewall-cmd --list-all -> it gives the current configuration
firewall-cmd --get-services -> Show a list of avaiable services
/usr/lib/firewalld/services -> directory of services
firewall-cmd --add-service smtp -> add a service

**** To make a modification, use firewall-cmd --reload. If we use the firewalll-cmd --add- service without the --permanent option, the rule will not be persistent after the reload. That is because the rules are written to the runtime configuration, with the --permanent, the rule is written to the permanent configuration.**

firewall-cmd --help | grep port -> options related to port configuration
firewall-cmd --add-port=123/tcp --permanent ; firewall-cmd --reload -> Open the 123 port and reload
man firewalld.richlanguage -> Man page for the rich rules with examples.

Example (Read the man pages above)

```
firewall-cmd --permanent --zone=testing --add-rich-rule='rule family=ipv4
source address=192.168.1.0/24 forward-port port=22 protocol=tcp
to-port=2222 to- addr=10.0.0.10'
```

Working with UFW

ufw status -> Status information ufw --help -> Help
ufw enable -> enable ufw
ufw allow ssh -> Open SSH port (Defined in /etc/services)
ufw reject out ssh -> reject outgoing ssh
ufw deny proto tcp from 192.168.4.10 to any port 22 -> Deny this host to access port 22
ufw app list -> Show app list
ufw allow Apache -> Allow apache
ufw logging on -> enable logging

Understanding X.509 Certificates and their Properties

Symmetric Encryption -> A secret key is used to encrypt and decrypt. Symmetric Encryption is fast

Algorithms:
→ DES
→ 3DES
→ Blowfish
→ AES

To create cipher text, there are some utilities:
→ GPG
→ openssl enc

Example:

openssl enc -des3 -salt -a -in /etc/passwd -out secret.des3 -> Create an encrypted file with DES3
openssl enc -d -des3 -salt -a -in secret.des3 -out unsecret -> Decrypt the file

Hash -> Is used to convert an input string with no fixed length to an ouput string of fixed length

Common Hash Algorithms
→ CRC-32
→ MD5
→ SHA1

Utillities to make hashes:

→ md5-sum
→ openssl dgst Example:
openssl dgst -sha1 /etc/passwd -> Create the hash
Another example:
/etc/passwd:

$6$/IIx6a0wmZRABy$JXYCEkb4YE7oGqPwisPu/
jzafAFLSK8wwUJVVf6Si1zXSfZGdZc6nuVpp5MbNU14ALfPB5RPk4YnjIFyjLE9p0
6 = Algorithm
IIx6a0wmZRABy$JXYCEkb4YE7oGqPwisPu = Salt
jzafAFLSK8wwUJVVf6Si1zXSfZGdZc6nuVpp5MbNU14ALfPB5RPk4YnjIFyjLE9p0
= Password

Asymmetric Encryption -> Public and private keys (Private used to encrypt and public to decrypt, slower, but more secure)

Used in:
-> Digital signatures
-> Key distribution
-> Digital certificates

Understanding Trust chains and public key infrastructures

In cryptography, X.509 is a standard defining the format of public key certificates. X.509 certificatesare used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web. They are also used in offline applications, like electronic signatures.

X509 is used to create PKI ceritficates. In PKI public and private keys are used for 3 different purposes:

→ Proof of identity
→ Encryption
→ Prove that the data is not tampered
In order to set a PKI, an certificate authority needs to be used.

The CA is a well known party that is used to sign the public key, resulting in a PKI certificate CSR(Certificate sign in request) -> Issued to a CA.

Extensions:
→ pem -> base64 certificate
→ der
→ cert/crt
→ p7b, p7c -> pkcs hash
→ p12 -> pkcs hash 12 extension
→ pfx

Generating and managing public and private keys

→ A public key infrastructure (PKI) is a set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public- key encryption. The purpose of a PKI is to facilitate the secure electronic transfer of information for a range of network activities such as e-commerce, internet banking and confidential email. It is required for activities where simple passwords are an inadequate authentication method and more rigorous proof is required to confirm the identity of the parties involved in the communication and to validate the information being transferred. [1]

→ In cryptography, a PKI is an arrangement that binds public keys with respective identities of entities (like people and organizations). The binding is established through a process of registration and issuance of certificates at and by a certificate authority (CA). Depending on the assurance level of the binding, this may be carried out by an automated process or under human supervision.

The trusted root signs a top level CA that signs another and so on.

openssl verify -verbose cert.pem -> Verify the chain of trust

Creating, operating and securing a certification authority

openssl genrsa -aes128 2048 > /etc/pki/tls/private/mykey.key -> generate an rsa key

openssl req -utf8 -new -key /etc/pki/tls/private/mykey.key -x509 -days 365 -out /etc/pki/tls/ certs/mykey.crt -set_serial 0 -> Create a pki certificate (Self Sign)

Request, Sign and manage server and client certificates

openssl genrsa -des3 -out root-ca.key 2048

openssl req -new -x509 -days 3650 -key root-ca.key -out root-cat.crt config openssl.cnf openssl x509 -noout -text -in root-ca.crt -> see the contents of the certificate (Now we can sign and copy our certificate)

Issuing a CSR

openssl genrsa 1024 > cert.key
openssl req -new -key cert.key -out cert.csr
openssl req -in cert.csr -noout -text -> see the content of the csr
openssl ca -in cert.csr -out cert.crt -> Ca that can be sent back to the server (Chain of trust)

Revoke Certificates and CA

CRL - Certificate Revocation List
openssl ca -revoke certificate -> Revoke option

Using The Openssl command for testing tls certificates

openssl s_client -connect ipa.example.com:636 -showcerts -> Tries to connect and show certificates on this host
genkey -> make easier to generate keys genkey --help -> Help
genkey --makeca server.example.com -> make a CA .

Performing Basic Gnupg Configuration, Usage and Key Revocation

gpg --gen-key -> Generate a gpg key
gpg --list-keys -> List gpg keys
gpg --export --armor joao > joao.key -> export the key
gpg --import joao.key -> import the key
gpg --keyserver pgp.mit.edu --send-keys keyid -> export the key to a key server

Using gnupg in Email

gpg --encrypt --armor --recipient joao message.txt -> Encrypt a file
gpg message.txt -> After the passwd, we can see the message

Using gpg to encrypt files

gpg --out encrypted-file --recipient joao --armor --encrypt somefile gpg --out textfile --decrypt somefile -> Decrypt the file.